Video-Like Lossless Compression of Data Cube for Big Data Query in Wireless Sensor Networks

RAY-I CHANG[1], YU-HSIEN CHU[1], CHIA-HUI WANG[2], NIANG-YING HUANG[1]
[1]Department of Engineering Science and Ocean Engineering, National Taiwan University,
No. 1, Sec. 4, Roosevelt Road, Taipei 10617, TAIWAN;
rayichang@ntu.edu.tw; d00525003@ntu.edu.tw; r99525049@ntu.edu.tw
[2]Department of Computer Science and Information Engineering, Ming Chuan University,
No. 5 Der-Ming Rd., Gwei Shan District, Taoyuan County City 333, TAIWAN
wangch@mail.mcu.edu.tw

Abstract: - Wireless Sensor Networks (WSNs) contain many sensor nodes which are placed in chosen spatial area to temporally monitor the environmental changes. As the sensor data is big, it should be well organized and stored in cloud servers to support efficient data query. In this paper, we first adopt the streamed sensor data as "data cubes" to enhance data compression by video-like lossless compression (VLLC). With layered tree structure of WSNs, compression can be done on the aggregation nodes of edge computing. Then, a parallel processing algorithm is designed to well organize and store these VLLC data cubes into cloud servers to support cost-effect big data query for machine learning and data mining. Our experiments are tested by real-world sensor data. Results show that our method can save 94% construction time and 79% storage space to achieve the same retrieval time in data query when compared with MySQL.

Key-Words: - Wireless sensor networks; Database storage; Video-like lossless compression; Parallel processing; Streamed sensor big data; Edge computing.

Received: February 22, 2021. Revised: July 28, 2021. Accepted: August 6, 2021. Published: August 10, 2021.

1 Introduction

Wireless sensor networks (WSNs) consist of small sensor nodes which have the capabilities of sensing, computation, and wireless communication [1]. They can be utilized for data collecting purposes in emergency applications such as environment monitoring, health monitoring, and structure monitoring [2][3][4]. Sensor nodes can be organized as an *ad hoc* network which can be widely used in human life. However, they are usually equipped with very limited resources of limited power and small storage. Researches show that, in a sensor node, the transmission of one single bit consumes the energy which is the same as the cost of executing more than one thousand operations in CPU [5]. Data compression of sensor data has become an important research topic [6][7].

Usually, two consecutive data that sensed by a same node would be similar. This indicates that the temporal correlation is preserved. Meanwhile, nearby sensors commonly gather the related data will also possess the spatial correlation of proximity. As mentioned in [7][8], spatial and temporal correlations are inherent in sensor data from most of sensory environment. Different compression methods were proposed to take advantages of temporal and spatial correlations to further extend life time of WSNs.

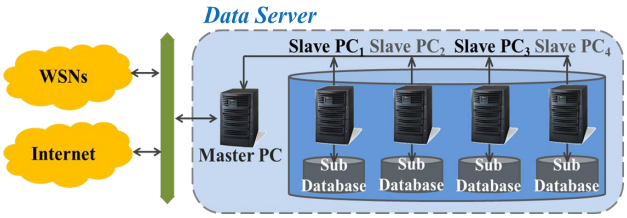


Fig. 1 Data server with parallel processing in database subsystems.

To promise cost-effective access, the data server is necessary to not only store sensor big data in resource-saving, but also provide capability of parallel processing of quick data query for customers. As shown in **Fig. 1**, the data server with parallel processing for sensor big data query is a composition of several slave PCs (peer computers) and their corresponding database subsystem. A master PC in data server plays the role of a supervisor. It determines the stored location of

query data and delivers commands to slave PCs to process their database subsystems.

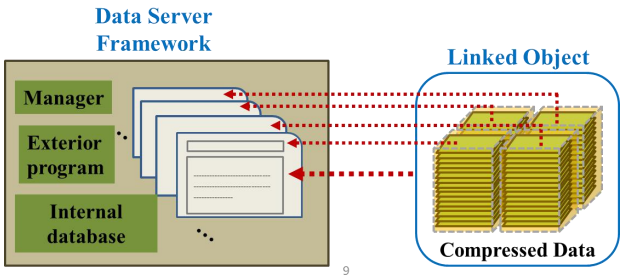


Fig. 2 Compressed sensor data is treated as linked object in data server.

In this paper, the data server treats the compressed sensor data as the linked object called "data cube" (as shown in **Fig. 2**) to couple video-like lossless compression (VLLC) to provide the cost-effective big data query for WSNs with edge computing. The remainder of this paper is organized as follows. In Section 2, the related works of WSNs data compression methods and data servers are introduced. In Section 3, we describe the proposed VLLC scheme with data cube storage structure for big data query in WSNs. Experiments in Section 4 demonstrate the performance evaluation of our proposed schemes. Conclusions and future works are shown in Section 5.

2 Related Works

There are two types of compression: lossy and lossless [9]. The lossy compression has better performance in compression ratio and energy-saving than lossless compression. But, the original streamed sensor data can't be recovered from lossy compression because of the data distortion. [10] applied an error-bounded lossy compression scheme [11] for high performance computing (HPC) and then used a machine learning technique to recover the streamed sensor data of wireless body sensor network (WBSN) within a given bounded error on the edge node. Different from [10], without the need to rebuild the lossy transmitted data on the edge node, [7] uses lossless compression schemes to consider data, spatial and temporal correlations among bounded-error-pruned sensor data. The data size in network transmitting and database storing are minimized for energy saving and to prolong the system lifetime.

In WSNs applications, customers can query the decompressed data from data server no matter the compression schemes applied were lossy or lossless. Therefore, data server is responsible for collecting and storing data sent from remote sensor nodes. It's also responsible for pre-processing of stored data (e.g. compressed data) and sending query results to customers. In data server, database subsystem can manage and organize the large amounts of sensor big data for customers' queries The databases such as Oracle [12], Microsoft SQL server [13], or MySQL [14], gather and organize the data stored in database server and provide query service for users' computers and applications. Since these databases are all needed to store big sensor data in cost-effectiveness, the above-mentioned compression schemes can be applied for these databases to reduce storage cost for big sensor data.

Notably, database provides operations for users to retrieve the collected data in many forms, and these data transactions may come up with several times of compression and decompression operations for data query. Meanwhile, users may not only focus in retrieving one data record but also a wide range of data records. Query for a wide range data may contain lots of sensor data and needs to perform lots decompressing processes. Therefore, this paper proposes a cost-effective compression scheme for big sensor data query to adapt trade-off between compression complexity and query performance for customers in prevalent WSNs applications.

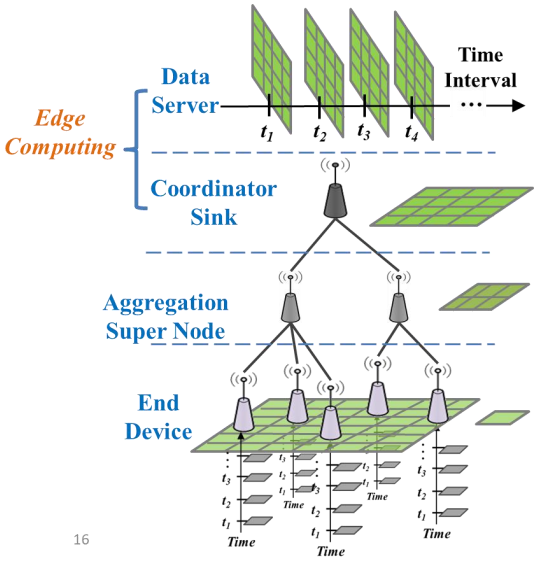


Fig. 3 Layered tree structure of WSNs with edge-computing VLLC

3 Video-Like Lossless Compression

In our proposed VLLC compression scheme, we apply a layered tree structure of WSNs as shown in **Fig. 3**. It includes three layers including the end-device of sensor nodes (with limited resources), the aggregation super node (with more resources for data aggregation and transmission of end-device), and the edge-computing coordinator sink (at the top

of WSNs to data aggregation of super nodes). Moreover, the coordinate sink usually wired with edge-computing data servers.

Both aggregation super nodes and coordinator sink are responsible for integrating the sensor data from different end devices at different sensor locations. Then, the edge-computing coordinator sink will use VLLC to compress the input stream data from aggregation super nodes and then forwards the compressed data steam to data server. The databases in data server will organize and manage the forwarded sensor data streams. It also allows customers to retrieve the stored data for different types of data queries in diversified WSNs applications.

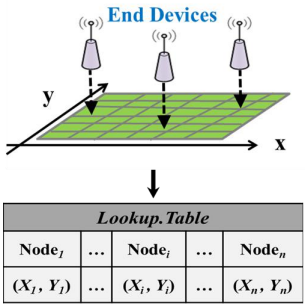


Fig. 4 Unique ID assignment by the location coordinate of end device.

The streamed sensor data from the end devices will be divided into several slices based on their recorded time intervals, and aggregation super nodes and edge-computing coordinator sink will integrate end-devices slices to a logical *plane*. To meet the logical plane requirement, each end device would need to be allocated with one unique ID. Using the coordinate system to map unique ID is a simple and fast way. As shown in **Fig. 4**, *Lookup.Table* is the mapping table for sensor data arrangement in the plane. Then, the edge-computing coordinator sink will use VLLC to compress these logical data planes of streamed sensor data and then send the compressed sensor data to data server for query later.

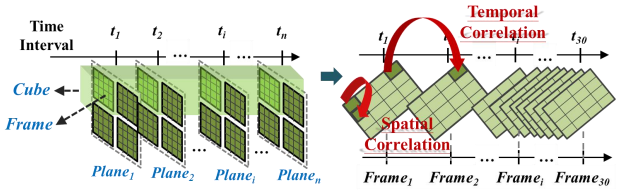


Fig. 5 Streamed sensor data structure applied in VLLC.

3.1 Proposed VLLC Scheme

As show in **Fig. 5**, a plane in a time interval is composed of many data slices from their end devices in the same interval. Data slices in a plane can be divided into small groups called frames. The cube is defined as the sequential frames from the same data-slice group in time period of sequential intervals. Therefore, the neighbouring data slices in a single frame indicate their end-devices are neighbors and they will preserve spatial correlations. Meanwhile, the data slices in the same locations on consecutive frames indicate the streamed sensor data came from the same end-device with temporal correlations. The inheritance of both temporal and spatial correlations in streamed sensor data from WSN motivates us to use VLLC for sensor big data query

The next important step for VLLC is streamed sensor data transformation to transform the raw streamed sensor data into formatted images. RGB color format is chosen in our VLLC method. One pixel is represented by 24 bits as the value of one sensor data. Each data is converted into 24 bits complied with RGB format, as shown **Fig. 6**.

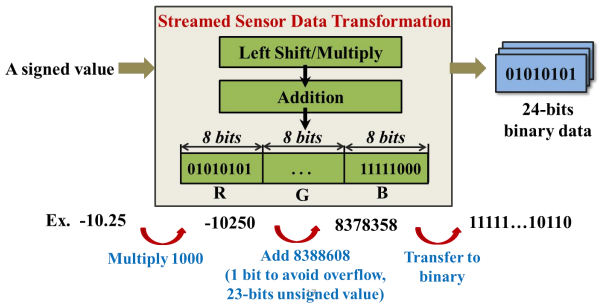


Fig. 6 Streamed sensor data transformation to RGB format.

The proposed architecture of data server provides parallel processing to several database subsystems The streamed sensor data from WSNs can be transformed to different cubes. As shown in the right hand side of **Fig. 5**, a cube of sensor data in a database subsystem, shows the GOP (group of pictures) is 30 frames as the default. After our streamed sensor data arrangement for VLLC, cubes of streamed sensor data are compressed separately in parallel, which can be maintained in different database subsystems. Furthermore, we adopt H.264 scheme in VLLC, because it is a standard of lossless video compression that can save large space effectively. In intra-frame compression of H.264, a single frame can be decompressed without refer to other frames. This important feature is helpful for the performance of data query.

3.2 Query Process in VLLC

All sensor data now are VLLC-compressed and stored in data server. The stored compressed data

cubes like several video clips are distributed in the database subsystems of data server. Query a single sensor data on data server is just like fetching a target frame of a specific video and the requested data can be simply obtained in the target frame. In **Fig. 7**, a single data query is performed in data server and request the $data_i$, which is recorded by $end\text{-}device_i$ at time t_i. The master PC in data server first check the location information of data from *Lookup.Table* for $node_i$, then forwards the query to a slave PC which contains the target data cube according the location information of $node_i$. The slave PC searches for the data cube in its database subsystem, then *Frame.offset* is set to the number of $Frame_l$ so as to pick out the specific frame. A specific pixel in this frame can be obtained after decompression, and the requested sensor data is obtained.

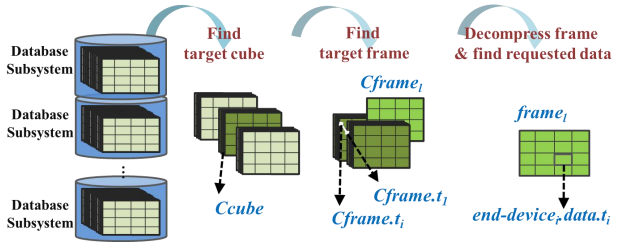


Fig. 7 Single data query in VLLC.

Users are interested in not only single sensor data from an end device, but also some range of data streams recorded from neighboring end-devices in a period of time. Range data query for an end device is similar to query sensor data in its own data cube in a database subsystem, but range query for neighboring end-devices will be involved.

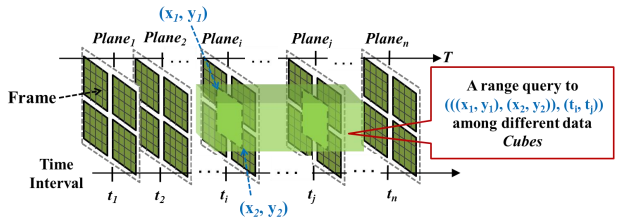


Fig. 8 Range data query in VLLC.

As an range query example shown in **Fig. 8**, range of (x_1,y_1) to (x_2,y_2) in a *plane* are residing in different data *cubes*, it means that the sensor data streams queried in this range are kept in different database subsystems of data server. The master PC first check the *Lookup.Table* for all end-devices, and determine corresponding slave PCs where the sub-queries are forwarded to. All the slave PCs involving in the query range are searching their own database subsystems, then set the *Frame.offset* from $Frame_i$ to $Frame_j$. These slave PCs then pick out the target *frames* and decompress the frames to the original sensor data. The master PC will merge the independent query results from slave PCs to the final range data query results for users.

3.3 Parallel processing in data server

Since operations for data modification and deletion is hardly occurred in WSN databases, the main performance issue on the WSN database can be associated with the processing time of data queries. Data server needs to provide query services for different users to make different query requests simultaneously. Since current computing devices usually support multi-core CPU, the decompression for different data cubes can be speed up by parallel processing to reduce the decompression time and increase the flexibility of data server for big sensor data query.

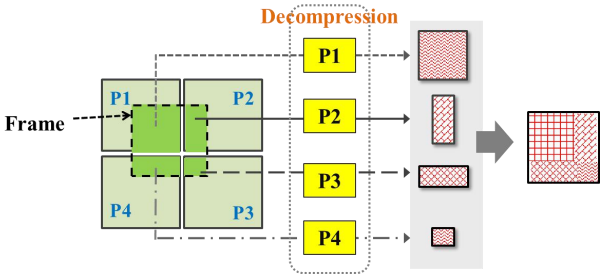
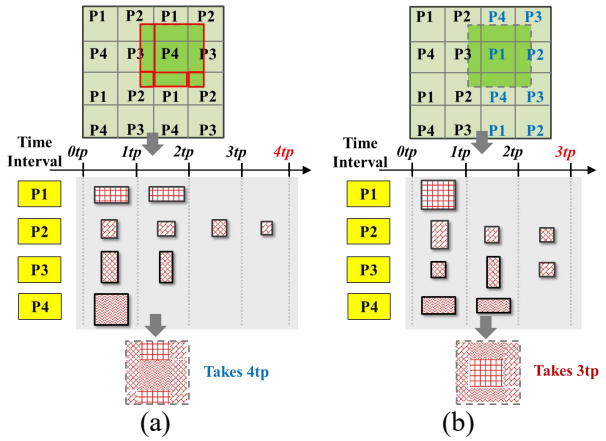


Fig. 9 Quad-cored parallel decompression work for frames in 4 data cubes.

Without loss of generality, we assume that our data server provides quad-core CPUs in all master and slave PCs. Upon the data query, the slave PC first sets the *Frame.offset* and the queried interval of time, then the following decompress work can be parallel. As an example shown in **Fig. 9**, the data query involves 4 different data cubes. Frames in 4 cubes can be evenly assigned to 4 processors (*i.e.* P1, P2, P3, P4) in quad-core CPU for decompression and transformation. However, if the number of data cubes in sensor data query is more than the process numbers (*i.e.* P1, P2, P3, P4) in slave PC, processor assignment is needed to optimize the data query performance.

As shown in **Fig. 10**(a), without processor assignment optimization, 9 data cubes with different sizes required to be decompressed in data query for a quad-core slave PC. We assume the time of decompress a single unit of data cube is t_p. P1 takes 2 t_p to finish its decompression work, P2 needs 4 t_p to finish, P3 needs 2 t_p and P4 takes 1 t_p. Overall decompression task costs 4 t_p time to complete the decompression work in parallel processing on quad-core slave PC.



(a) (b)

Fig. 10 Parallel processing of decompression work for frames in 9 data cubes. (a) Without improvement. (b) With improvement.

After improvement, 4 processors will takes 1 t_p , 3 t_p, 3 t_p and 2 t_p separately for the same query as shown in **Fig. 10**(b). The total compression time is 3 t_p, and it takes 1 t_p less than the previous processor assignment. We can have an improved performance for completing the data query task by reordering the data-cube decompression sequence of processor assignment.

4 Experiments

In our experiments, the dataset is real-world data [15] of daily global air temperature monitored during 1950 to 1999. It contains both spatial and temporal correlations with 8,190 sensor nodes in a continuous range of latitude and longitude. Each node monitored 177,380 days continuously. There are totally 145,274,220 records of streamed sensor data. We compare the performance metrics of space savings, compression time and query time of data cubes with different frame sizes.

To evaluate the performance in VLLC-enable data server when sensor data are divided into different frame resolutions (*i.e.* different number of data cubes). As shown in **Fig. 11**, they have 1x1, 2x2 and 4x4 *Cubes* respectively. We split the raw dataset into three different sizes for different frame resolutions, which are 90x91, 180x182 and 360x364. Due to the split on the same raw dataset, the number of *frames* in a cube is in inverse proportion to the *Frame* size. The *Cube* number can be taken as the number of database subsystems. It also indicates the parallel processing ability of query in data server.

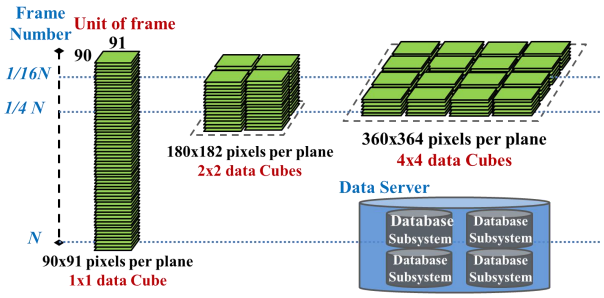


Fig. 11 Different frame resolution settings for applied dataset.

Our VLLC uses video codec of x264 [16] to compress these *Cubes* built from raw dataset, and use FFmpeg [17] software to do VLLC decompression. The performance of different frame resolutions in data cubes with different sizes is shown in **Table 1**. The ratio of space savings is calculated by Eq. (1).

$$Space\_Savings = 1 - \frac{Compressed\_Data\_Size}{Original\_Data\_Size} \quad (1)$$

Table 1 Performance of different frame resolutions.

	Frame Size (*m* by *n*) (Pixels)		
	90 x91	***180 x182***	***360 x364***
Applied Dataset Size	5568.768MB		
Number of Cubes	1x1	2x2	4x4
Total Frame Number	17738	4435	1109
Database Construct Time	980.794s	866.586s	827.819s
Original Data Size	420MB	417MB	415MB
Compressed Data Size	206MB	198MB	195MB
Compression Time	53.02s	39.22s	33.43s
Space Savings	96.30%	96.44%	96.50%
Single Frame Decompression Time	0.01(s)	0.125(s)	0.385(s)

As shown in **Fig. 12**, *Frames* with larger sizes achieve better performance in both of compression and construction time. They are dependent on the number of frames in *data cubes*.

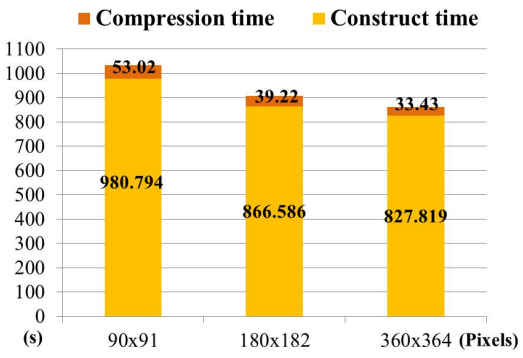


Fig. 12 Compression and construction time for different frame resolutions in data cubes.

Decompress single *Frame* from a small-sized *data cube* would cost less time than the large one are shown in **Fig. 13**. The single data query performance for different frame resolutions in different sizes of data cubes is shown in **Fig. 14**.

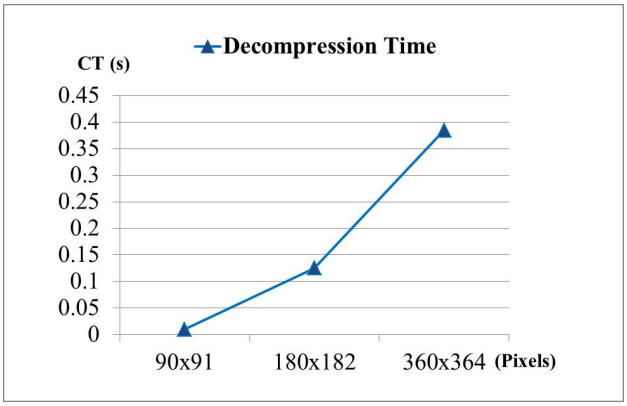


Fig. 13 Decompression time of single frame in different-sized data cubes.

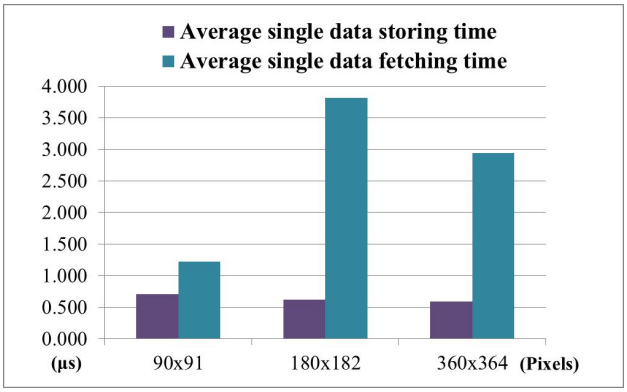


Fig. 14 Average time for single data storing and fetching via different numbers of data cubes.

In the second experiment, we compare our VLLC-enabled data server with a well-known open-source database MySQL (v5.5). In this experiment, we use the same real-world dataset as used in previous experiment and they are stored in only one database for each scheme. These two databases from different schemes are need to construct 145,274,220 counts of sensor data, and they are needed to be retrievable. The cost of storage space and construct time is shown in **Table 2.**

Table 2 Performance of MySQL and VLLC.

	WSN Data Server Scheme	
	MySQL	**VLLC**
Sensory Data Size	5568.768MB	
Database Size	946.8MB	206.0MB
Space Savings	83.00%	96.30%
Construct Time	17121.000(s)	1033.814(s)

We assume the MySQL performance results as the comparison base for performance metrics of database size, construction time and space saving. Our VLLC method saves approximately to 94% of construction time compared to MySQL. Since our method is not a relational database model, VLLC still promote the space saving ratio to 96.30%. It is over 13% more than conventional MySQL. After normalization, the storage cost of VLLC-enabled database is 79% less than MySQL.

Table 3 shows the experiment result in the query performance of our method and MySQL. The unit of retrieve data means how many data will involve in a query at a time. In our method, the value of unit data is binding to the frame size (90x91, 180x182 or 360x364). When we aim to retrieve the data in a database with an unit data with size 90x91, it is equal to retrieve size 8190 (=90x91) data in MySQL. Experiment results show that our method takes 0.01 seconds to retrieve size 8190 (=90x91) data, and MySQL need to take 0.022 seconds. It is over double time.

Table 3 Data Retrieval Time of MySQL and VLLC.

		Total retrieve data		
Database	**Unit of Retrieve Data**	**8190** =90x91	**32760** =90x91x4	**131040** =90x91x16
VLLC	90x91	0.010s	0.040s	0.160s
MySQL	Any	0.022s	0.071s	0.351s

Estimated time

Note that the request for retrieve a frame with the size of 180x182 can be achieved by retrieve 4 continues frames with size of 90x91. The estimated time can be concerned as the evaluation for each query request. It means that our method can achieve the performance of data query as well as the MySQL or even better than.

5 Conclusion and Future Works

A video-like lossless compression method for big sensor data query is proposed to manage huge amount of WSNs data. The compression method geographically treat streamed data as many data cubes with temporal and spatial correlations. The data cubes can be distributed to different database subsystems in data server for load-balancing and parallel processing of the data queries. Experimental results demonstrate that the trade-offs between space saving and query time can be determined by adopting different sizes of data cube. The proposed VLLC method achieves much better performance than MySQL in construct time and storage saving. VLLC saves approximately 94% of construct time compared to MySQL, and the storage cost of VLLC

is 79% less than MySQL.

Range query plays an important role in common database operations. It provides users a convenient way to request a large number of data for machine learning and data mining [18]. There are future works to enhance the performance of range query in our system. (1) Cache the common requested data on edge servers to reduce query time. (2) Duplicate the database or data cubes to have better query performance. (3) Improve the compression ratio by using other video codecs such as H.265 and H.266.

References:

[1] C.F. García-Hernández, P.H. Ibargüengoytia-González, J. García-Hernández, and J.A. Pérez-Díaz, "Wireless sensor networks and applications: a survey", *International Journal of Computer Science and Network Security*, Vol.7, pp. 264-273, March 2007.

[2] D. Culler, D. Estrin, and M. Srivastava, "Guest editors' introduction: overview of sensor networks", *Computer*, Vol.37, pp. 41-49. August 2004.

[3] Chu, Yu-Hsien, *et al.* "UPHSM: Ubiquitous personal health surveillance and management system via WSN agent on open source smartphone," 2011 IEEE 13th International Conference on e-Health Networking, Applications and Services. IEEE, 2011.

[4] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson and D. Culler, "An analysis of a large scale habitat monitoring application," Embedded Networked Sensor Systems, November 2004.

[5] Y. Liang and W. Peng, "Minimizing energy consumptions in wireless sensor networks via Two-Modal Transmission", *ACM SIGCOMM*, Vol.40, pp. 12-18, January 2010.

[6] K.C. Barr and K. Asanovic, "Energy-aware lossless data compression," ACM Transactions on Computer Systems, vol. 24, pp. 250-291, August 2006.

[7] Chang, Ray-I., *et al.* "Bounded-error-pruned sensor data compression for energy-efficient IoT of environmental intelligence." Applied Sciences 10.18 (2020): 6512.

[8] Z. Zhang and O. Berger, "Cluster based data query analysis and optimization for Wireless Sensor Networks," Advanced Communication Technology, February 2008.

[9] Hamdan, Salam, Arafat Awaian, and Sufyan Almajali. "Compression techniques used in IoT: a comparitive study," 2019 2nd International Conference on new Trends in Computing Sciences (ICTCS). IEEE, 2019.

[10] Azar, Joseph, *et al.* "An energy efficient IoT data compression approach for edge machine learning." Future Generation Computer Systems 96 (2019): 168-175.

[11] Di, Sheng, and Franck Cappello. "Fast error-bounded lossy HPC data compression with SZ." 2016 IEEE international parallel and distributed processing symposium (IPDPS). IEEE, 2016.

[12] Oracle, Available: http://www.oracle.com/tw/index.html.

[13] SQL Server, Available: http://www.microsoft.com/taiwan/sql/default.mspx

[14] MySQL: The world's most popular open source database, Available: http://www.mysql.com

[15] SensorKDD-2009 Challenge, Available: http://www.ornl.gov/sci/knowledgediscovery/SensorKDD-2009/challenge.htm

[16] x264 Home Page, VideoLan Organization, Available: http://www.videolan.org/developers/x264.html

[17] FFmpeg, Available: http://www.ffmpeg.org/.

[18] Yu-Hsin Hung, Yi-Jie Wang, Ray-I Chang, "Investigation of the effective use of ensemble learning algorithms for cyber data analytics," ACM International Conference on e-Society, e-Education and e-Technology (ICSET). ACM, 2020.